项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 王星洲

学 号： 1652977

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc531701909)

[1.1 背景分析 1](#_Toc531701910)

[1.2 功能分析 1](#_Toc531701911)

[2 设计 1](#_Toc531701912)

[2.1 数据结构设计 1](#_Toc531701913)

[2.2 队列的使用 1](#_Toc531701914)

[2.3 流程设计 1](#_Toc531701915)

[3 实现 2](#_Toc531701916)

[3.1链表的创建 2](#_Toc531701917)

[3.1.1 主要代码 2](#_Toc531701918)

[3.1.2 注意事项 2](#_Toc531701919)

[3.2 接收用户的输入 3](#_Toc531701920)

[3.2.1用户输入过程流程图 3](#_Toc531701921)

[3.2.2核心代码 4](#_Toc531701922)

[3.2.3 人机交互截屏示例 5](#_Toc531701923)

[3.2.4 人机交互优势及注意事项 5](#_Toc531701924)

[3.3 队列输出 8](#_Toc531701925)

[3.3.1 队列输出功能流程图 8](#_Toc531701926)

[3.3.2 队列输出功能核心代码 9](#_Toc531701927)

[3.3.3 队列输出功能亮点及注意事项 10](#_Toc531701928)

[4 测试 10](#_Toc531701929)

[4.1 功能测试 10](#_Toc531701930)

[4.1.1 一般功能测试(A窗口人多) 10](#_Toc531701931)

[4.1.2一般功能测试(B窗口人多) 10](#_Toc531701932)

[4.2 边界测试 11](#_Toc531701933)

[4.2.1 最小值N测试(A) 11](#_Toc531701934)

[4.2.1 最小值N测试(B) 11](#_Toc531701935)

[4.3 出错测试 12](#_Toc531701936)

[4.3.1 输入数据非法 12](#_Toc531701937)

[4.3.2 输入N值超出范围 12](#_Toc531701938)

[4.3.3 number非法之后后续输入 12](#_Toc531701939)

[4.3.4 输入数据超过需求 12](#_Toc531701940)

[5 亮点 13](#_Toc531701941)

[5.1队列输出算法亮点 13](#_Toc531701942)

[5.2人机交互亮点 13](#_Toc531701943)

# 1 分析

## 背景分析

设某银行有A，B两个业务窗口，且处理业务的速度不一样，其中A窗口处理速度是B窗口的2倍----即当A窗口每处理完2个顾客是，B窗口处理完1个顾客。给定到达银行的顾客序列，请按照业务完成的顺序输出顾客序列。假定不考虑顾客信后到达的时间间隔，并且当不同窗口同时处理完2个顾客时，A窗口的顾客优先输出。

## 1.2 功能分析

输入说明：输入为一行正整数，其中第一数字N（N<=1000）为顾客总数，后面跟着N位顾客的编号。编号为奇数的顾客需要到A窗口办理业务，为偶数的顾客则去B窗口。数字间以空格分隔。

输出说明：按照业务处理完成的顺序输出顾客的编号。数字键以空格分隔，但是最后一个编号不能有多余的空格。

# 2 设计

## 2.1 数据结构设计

本题数据为先入先出型，所以考虑使用队列解决问题。我们假设顾客进入银行按照自己的序号自动排成了两个队列qA和qB，再按照顺序离开银行，就可以得到所求的序列了。因此，我们使用qA，qB两个队列，接收用户输入的int值，即可实现需求。

## 2.2 队列的使用

采用#include<queue>语句来使用STL提供的标准队列，在向队列中添加元素时，使用push()函数将数值添加到队列末尾，在访问数据时，使用front()查询队首的值，输出之后使用pop()函数将其移出队列。

## 2.3 流程设计

本程序首先使用N接收用户输入的顾客数量，再用number接收用户输入的顾客编号，编号允许重复使用，接收用户输入时会加入输入检测，保证程序流畅运行。之后根据用户输入的值用A或B队列存储该值，奇数进入A队列，偶数进入B队列，之后按照A输出两个B输出一个的顺序，利用循环将两队列中储存的数值输出，得到结果。

# 3 实现

## 3.1链表的创建

### 3.1.1 主要代码

queue<int> qA;

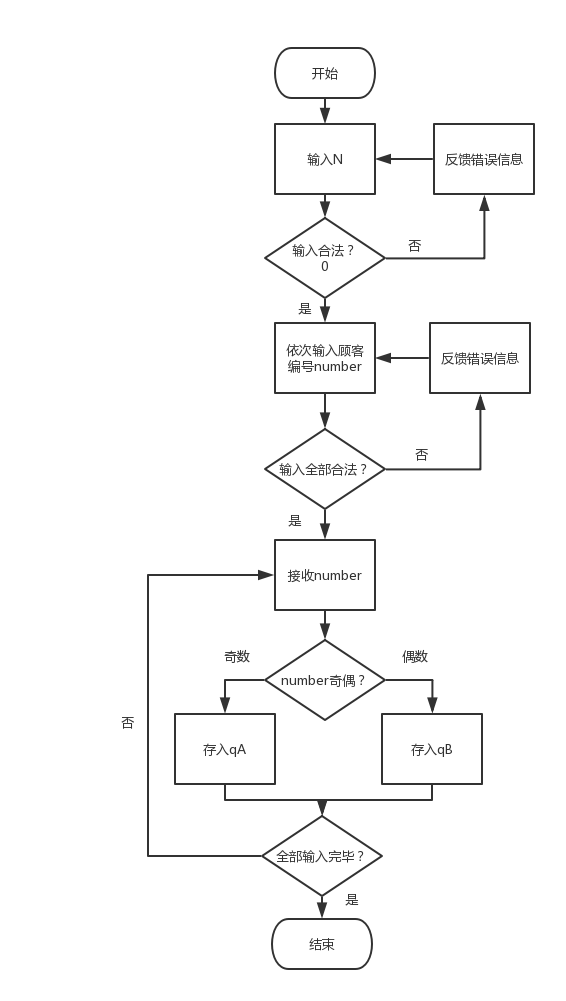
queue<int> qB;

### 3.1.2 注意事项

1.注意包含头文件<queue>

## 3.2 接收用户的输入

### 3.2.1用户输入过程流程图



### 3.2.2核心代码

//接收用户的输入

int N, number;

//N的处理

while(1)

{

//输入非法

if (!(cin >> N))

{

cout << "输入失败，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

//输入负数或0

else if (N <= 0)

{

cout << "请输入正整数。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

//输入的值超过题目要求

else if (N > 1000)

{

cout << "请输入不超过1000个数。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

//输入正确

else

{

break;

}

}

//对于number的处理

for (int i = 0; i < N; i++)

{

//输入非法

while(!(cin >> number))

{

cout << "输入失败，已经知道您要输入" << N << "个顾客的编号，请重新输入编号。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

while (!qA.empty())

{

qA.pop();

}

while (!qB.empty())

{

qB.pop();

}

i = 0;

}

//奇数，加入A队列

if (number % 2 != 0)

{

qA.push(number);

}

//偶数，加入B队列

else

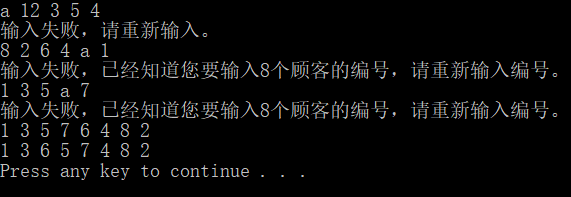
{

qB.push(number);

}

}

### 3.2.3 人机交互截屏示例



### 3.2.4 人机交互优势及注意事项

1.对于N，我在输入中添加了合法检测

//输入非法

if (!(cin >> N))

{

cout << "输入失败，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

//输入负数或0

else if (N <= 0)

{

cout << "请输入正整数。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

//输入的值超过题目要求

else if (N > 1000)

{

cout << "请输入不超过1000个数。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

}

因为N是int类型，所以输入不合法的值会进入if语句，提示用户重新输入，同时清空输入缓存区，更改cin标志，为下次输入做准备，这样做大大提升了用户的体验，防止程序出现问题。

2. 对于number，我在输入中添加了合法检测

//对于number的处理

for (int i = 0; i < N; i++)

{

//输入非法

while(!(cin >> number))

{

cout << "输入失败，已经知道您要输入" << N << "个顾客的编号，请重新输入编号。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

while (!qA.empty())

{

qA.pop();

}

while (!qB.empty())

{

qB.pop();

}

i = 0;

}

当有number输入出错时，清空队列以及输入流，将i置回0，重新接收number的输入，保证用户使用的流畅。

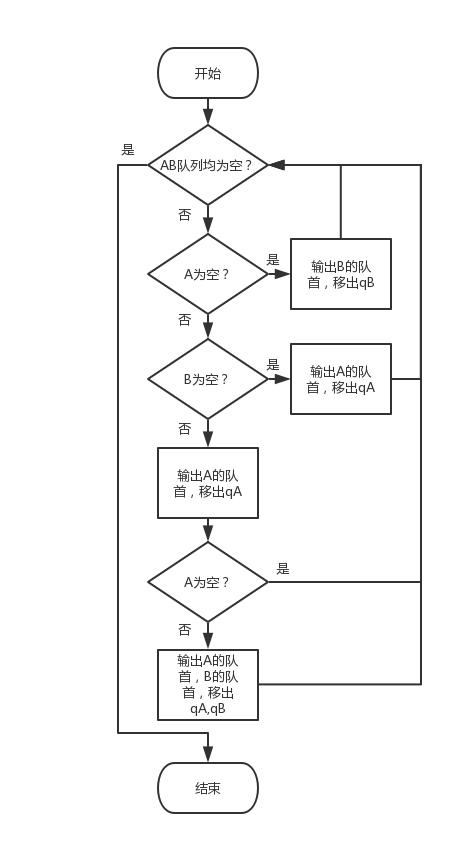
3.对于两个队列，当number为奇数，存入A队列，当number为偶数，存入B队列。

4.注意，检测合法代码中需要将输入流清空，否则将会在输出语句出现死循环

5.number输入出错中务必将两队列清空。否则会让之前的错误数据污染队列。

## 3.3 队列输出

### 3.3.1 队列输出功能流程图



### 3.3.2 队列输出功能核心代码

//当A或者B不为空时，进入循环

while (!qA.empty() || !qB.empty())

{

//如果队列A空，则将B按序全部输出

if (qA.empty())

{

cout << qB.front() << " ";

qB.pop();

}

//如果队列B空，则将A按序输出

else if (qB.empty())

{

cout << qA.front() << " ";

qA.pop();

}

//两者都不为空时

else

{

//输出A的第一个

cout << qA.front() << " ";

//将该值移出队列

qA.pop();

//如果此时A已经空，则返回循环起始

if (qA.empty())

{

continue;

}

//若A仍不为空，则再输出一个A然后输出B

else

{

cout << qA.front() << " ";

qA.pop();

cout << qB.front() << " ";

qB.pop();

}

}

}

//消去最后的空格

cout << "\b";

### 3.3.3 队列输出功能亮点及注意事项

1.在输出了一个A以后请务必判断A是否为空，不然会导致输出错误结果。

2.使用队列需要注意pop()函数，不会提供返回值，所以，不可以使用cout<<qA.pop()语句来进行输出。

3.在输出结束后会有一个多余的空格，应用\b消去该空格以符合题意。

# 4 测试

## 4.1 功能测试

### 4.1.1 一般功能测试(A窗口人多)

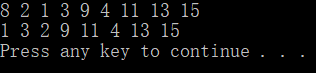
**测试用例**：

输入 8 2 1 3 9 4 11 13 15

**预期结果**：

输出1 3 2 9 11 4 13 15

**实验结果**



### 4.1.2一般功能测试(B窗口人多)

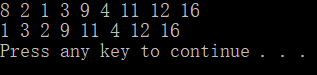
**测试用例：**

输入 8 2 1 3 9 4 11 12 16

**预期结果：**

输出 1 3 2 9 11 4 12 16

**实验结果：**



## 4.2 边界测试

### 4.2.1 最小值N测试(A)

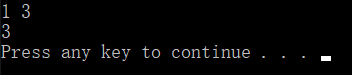
**测试用例：**

输入 1 3

**预期结果：**

输出3

**实验结果：**



### 4.2.1 最小值N测试(B)

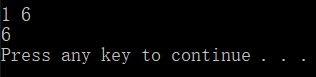
**测试用例：**

输入 1 6

**预期结果：**

输出6

**实验结果：**



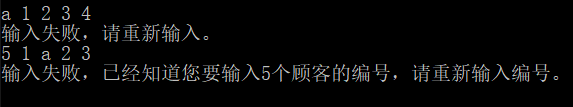
## 4.3 出错测试

### 4.3.1 输入数据非法

**测试用例：**输入非int

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

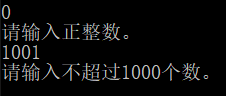


### 4.3.2 输入N值超出范围

**测试用例：**输入 0， 1001

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

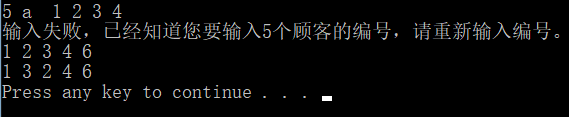


### 4.3.3 number非法之后后续输入

**测试用例：**number输入非法，之后输入一段合法序列

**预期结果：**程序给出提示信息，程序正常运行不崩溃，之后正常工作。

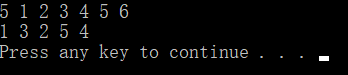
**实验结果：**



### 4.3.4 输入数据超过需求

**测试用例：**输入5 1 2 3 4 5 6

**预期结果：**根据题目要求，之将前五个数据存储输出。输出 1 3 2 5 4

**实验结果：**

# 5 亮点

### 5.1队列输出算法亮点

参考3.3.3

### 5.2人机交互亮点

参考3.2.4